Photon Cloud(三)
雙向傳輸

主講：紀曲峰

基礎知識

- 一般來說，線上遊戲的角色都是動態產生，包含自己的
- 動態產生的資源必須放在Resources資料夾底下，若有動態建立時PUN將自動從Resources資料夾下搜尋後產生物件

加上登入場景

建立登入用場景

- 建立場景讓玩家可以取名字用，以免名字全部變成player1,player2....
- 將之前的場景複製一份，取名為Stage01
- 開一個新場景，儲存名稱為MenuStage

新增Menu腳本

- 建立腳本取名為PhotonMenu，並修改內容如下

```
[RequireComponent(typeof(ConnectAndJoinRandom))]
public class PhotonMenu : MonoBehaviour {

}
```

加入變數

- 加入三個變數
  playerName : 玩家名字
  autoJoin : 自動登入的腳本
  ErrorMessage : 登入時的錯誤訊息

```
private string playerName = "";
private ConnectAndJoinRandom autoJoin;
private string ErrorMessage = "";
```

加入初始化

- 初始化時讀取ConnectAndJoinRandom，並將自動登入的開關設為false，等到玩家取好暱稱後才重新開啟自動登入

```
void Awake () {
    autoJoin = GetComponent<ConnectAndJoinRandom> ();
    autoJoin.AutoConnect = false;
}
```

加入UI

- 加上UnGUI讓玩家輸入名字，並加上按鈕，若按下按鈕則將PhotonNetwork.playerName傳入玩家姓名，然後將自動登入autoJoin.AutoConnect設為true

```
void OnGUI () {

    GUILayout.Label("Player name :");
    playerName = GUILayout.TextField(playerName);
    if (GUILayout.Button("Join Game", GUILayout.Width(100)))
    {
        ErrorMessage = "";
        if( playerName.Length > 0 )
        {
            PhotonNetwork.playerName = playerName;
            autoJoin.AutoConnect = true;
        }
        else
        {
            ErrorMessage = "You must input a name.";
        }

    }

    if (ErrorMessage.Length > 0)
        GUILayout.Label (ErrorMessage);

}
```

加入登入後事件

- 加入OnJoinedRoom()事件接收登入的回傳，當登入完成後呼叫PhotonNetwork.LoadLevel()開啟遊戲場景

- 切換場景請使用PhotonNetwork.LoadLevel()以保留Photon所需變數或資料

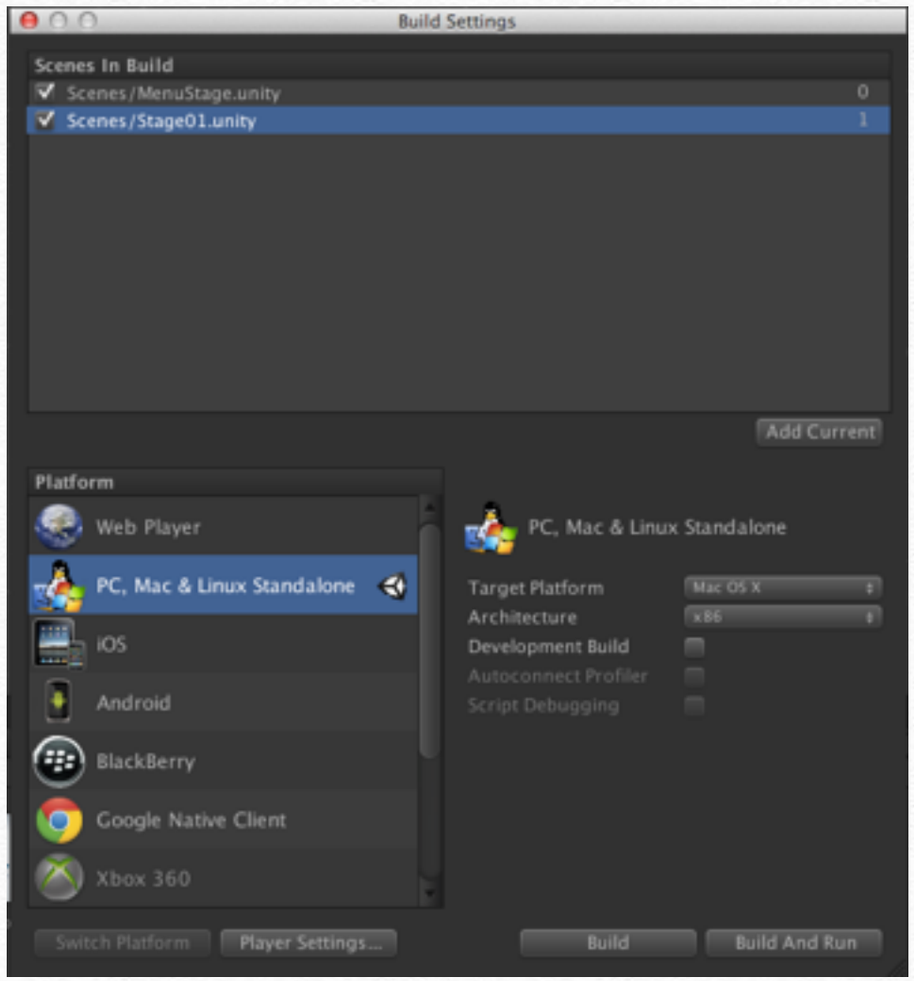
```
public void OnJoinedRoom()
{
    Debug.Log("OnJoinedRoom");
    PhotonNetwork.LoadLevel("Stage01");
}
```

為場景加上Menu腳本

- 建立一個空物件，命名為PhotonObj，然後將腳本PhotonMenu放到此物件上

設置要編譯的場景

- 到Build Settings視窗加入MenuStage和Stage01兩個場景



執行Menu場景

- 執行後遊戲，已可登入並切換場景了

移除Stage01的登入項目

移除遊戲場景的登入

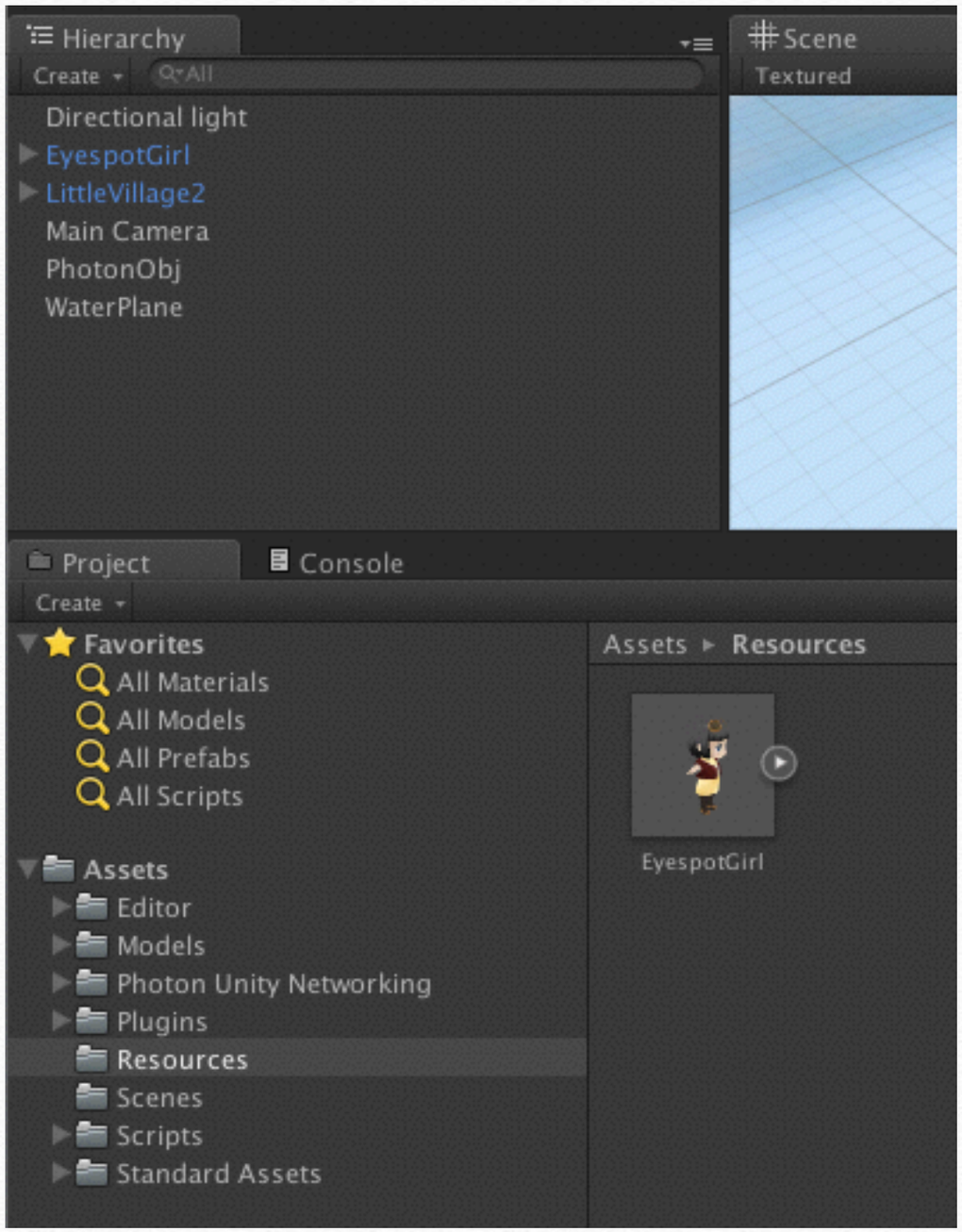
- 原本的Stage01的PhotonObj也有一個自動登入，請把他移除

- 重新執行MenuStage進入遊戲，使用聊天室時已可以用自己取的暱稱顯示

製作角色Prefab

建立角色Prefab

- 建立Resources資料夾
- 在Resources內新增Prefab，並將主角拉進去
- 完成後將場景中的角色刪除



動態建立角色

新增腳本

- 建立C#腳本，取名為CharacterInGame
- 修改繼承為Photon.MonoBehaviour

```
using UnityEngine;
using System.Collections;

public class CharacterInGame : Photon.MonoBehaviour {


}
```

公用變數

- 新增一個公用變數來設定角色模型，若是玩家自訂角色可自行修改為動態讀取

```
    public Transform playerPrefab;
```

建立角色

- 在PhotonCloud裡建立角色非常簡單，只要在場景一開始建立好自己的角色，不需建立其他玩家的，PUN會自動同步所有玩家角色

- 需要同步的角色Prefab(如玩家角色或敵物)一定要放在Resources資料夾內，PUN會自動讀取並在場景中建立角色

建立角色的程式

- 在Awake裡用PhotonNetwork.Instantiate指令建立角色
- 加入判斷若未成功連線自動回到登入場景登入

```
    public void Awake()
    {
        // in case we started this demo with the wrong scene being active, simply load the menu scene
        if (!PhotonNetwork.connected)
        {
            Application.LoadLevel("MenuStage");
            return;
        }

        // we're in a room. spawn a character for the local player. it gets synced by using PhotonNetwork.Instantiate
        PhotonNetwork.Instantiate(this.playerPrefab.name, transform.position, Quaternion.identity, 0);
    }
```

當Master離開遊戲

- 若開局者離開遊戲時，Master會自動切換為下一個順位的進入者，並且觸發OnMasterClientSwitched事件

- 接收到事件後可以選擇解散遊戲或由新的Master接手繼續遊戲

- 若要做接手機制必須將所有動態產生的敵物或事件以全域變數維護，才不會隨著Master的離開而消失

若有玩家離開遊戲

- 當有玩家離開遊戲時會觸發
  OnPhotonPlayerDisconnected(PhotonPlayer player)
- 有些遊戲人數一定要足才能進行，例如棋類遊戲，此時可以解散遊戲

Master離開的事件處理

- 此處用了一個技巧，若有掛入聊天室則將訊息顯示在聊天室

```
    public void OnMasterClientSwitched(PhotonPlayer player)
    {
        Debug.Log("OnMasterClientSwitched: " + player);

        string message;
        InRoomChat chatComponent = GetComponent<InRoomChat>();  // if we
find a InRoomChat component, we print out a short message

        if (chatComponent != null)
        {
            // to check if this client is the new master...
            if (player.isLocal)
            {
                message = "You are Master Client now.";
            }
            else
            {
                message = player.name + " is Master Client now.";
            }

            chatComponent.AddLine(message); // the Chat method is a RPC.
as we don't want to send an RPC and neither create a PhotonMessageInfo,
lets call AddLine()
        }
    }
```

當有玩家進入的事件處理

- 當有玩家進來時將訊息顯示在聊天室

```
public void OnPhotonPlayerConnected(PhotonPlayer player)
{
    Debug.Log("OnPhotonPlayerConnected: " + player);

    string message;
    InRoomChat chatComponent = GetComponent<InRoomChat>();

    if (chatComponent != null)
    {
        message = "OnPhotonPlayerConnected: " + player;
        chatComponent.AddLine(message);
    }
}
```

當玩家離開時的事件

- 當玩家離開時將事件顯示在聊天室

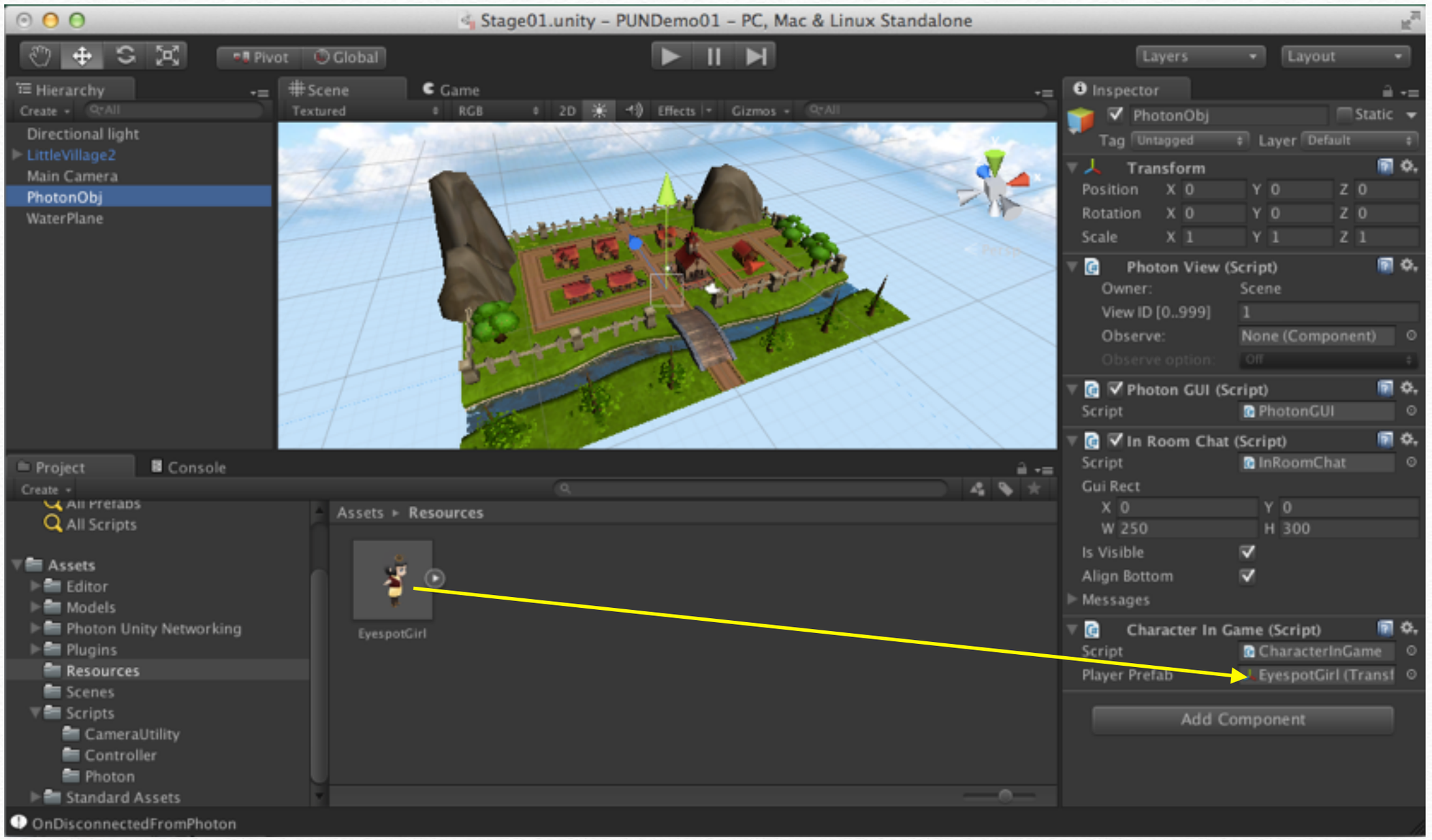
```
public void OnPhotonPlayerDisconnected(PhotonPlayer player)
{
    Debug.Log("OnPlayerDisconneced: " + player);

    string message;
    InRoomChat chatComponent = GetComponent<InRoomChat>();

    if (chatComponent != null)
    {
        message = "OnPlayerDisconneced: " + player;
        chatComponent.AddLine(message);
    }
}
```

其他事件

```
    public void OnLeftRoom()
    {
        Debug.Log("OnLeftRoom (local)");

        // back to main menu
        Application.LoadLevel("MenuStage");
    }

    public void OnDisconnectedFromPhoton()
    {
        Debug.Log("OnDisconnectedFromPhoton");

        // back to main menu
        Application.LoadLevel("MenuStage");
    }

    public void OnPhotonInstantiate(PhotonMessageInfo info)
    {
        Debug.Log("OnPhotonInstantiate " + info.sender);    // you could use
this info to store this or react
    }

    public void OnFailedToConnectToPhoton()
    {
        Debug.Log("OnFailedToConnectToPhoton");

        // back to main menu
        Application.LoadLevel("MenuStage");
    }
```

將腳本加到場景

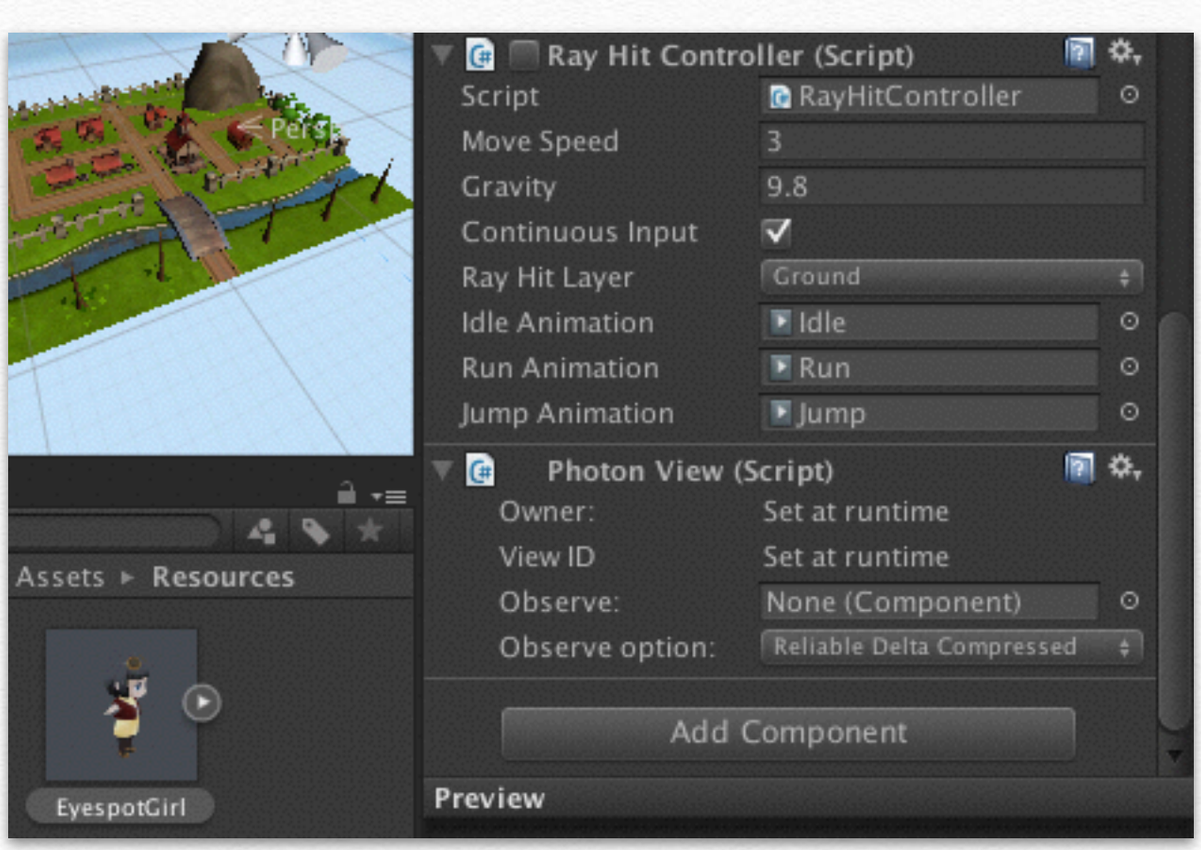
- 將腳本加到PhotonObj並將角色Prefab拉到playerPrefab變數



加入角色的處理腳本

移除舊腳本

- 點選角色Prefab，若PhotonMoverStream或PhotonMover還在角色身上請將他移除



建立腳本

- 建立一個腳本，取名為MoveNetWorking

```
using UnityEngine;
using System.Collections;

public class MoveNetWorking : Photon.MonoBehaviour {

}
```

加入變數

- 加入需要的變數，因為我的攝影機會跟隨玩家，因此需讀入攝影機處理用的腳本 CameraOrbit
- namePosition用來計算玩家名字從3D座標轉換到GUI座標用的

```
private Vector3 correctPlayerPos = Vector3.zero; // 移動後的位置
private Vector3 correctPlayerRot = Vector3.zero; // 移動後的方向

private RayHitController rayController;
private string sendAniName = "";          // 發送的動畫
private string receiveAniName = "";          // 接收的動畫
private CameraOrbit cameraOrbit;
private Vector3 namePosition;

private string characterName;             // 角色暱稱
```

初始化

```
    void Awake () {
        correctPlayerPos = this.transform.position;
        correctPlayerRot = this.transform.eulerAngles;
        rayController = this.GetComponent<RayHitController> ();
        cameraOrbit = Camera.main.GetComponent<CameraOrbit> ();

        characterName = photonView.owner.name;

        if (photonView.isMine)
        {
            //MINE: local player, simply enable the local scripts
            rayController.enabled = true;
            cameraOrbit.target = this.transform;
        }
        else
        {
            rayController.enabled = false;
        }

        gameObject.name = gameObject.name + photonView.viewID;

    }
```

內容說明

- 無論是我方或其他玩家角色，任何角色建立時都會觸發此腳本
- 取得名字用photonView.owner.name，若用PhotonNetwork.playerName只能取得登入者的名字，無法取得其他玩家名字
- 判斷此角色是否為玩家自己，可使用 if(photonView.isMine) { … }

加入Serialize傳輸

- Serialize傳輸內容和之前的一樣

```
void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    if (stream.isWriting) {
        stream.SendNext(transform.position);
        stream.SendNext(transform.eulerAngles);
        stream.SendNext(sendAniName);
    }
    else {
        correctPlayerPos = (Vector3)stream.ReceiveNext();
        correctPlayerRot = (Vector3)stream.ReceiveNext();
        receiveAniName = (string)stream.ReceiveNext();
    }
}
```

加入Update

- 若不是自己則利用Lerp將角色滑到目標並播放動畫，否則只更新動畫名稱供廣播用，最後用WorldToScreenPoint取得角色對應到GUI座標位置。

```
void Update () {
    float moveSpeed = 5;

    if (!photonView.isMine)
    {
        transform.position = Vector3.Lerp(transform.position,
correctPlayerPos, Time.deltaTime * moveSpeed);
        transform.eulerAngles = correctPlayerRot;
        if( receiveAniName.Length > 0 )
        {
            this.animation.Play (receiveAniName);
        }
    }
    else
    {
        sendAniName = rayController.CharacterAniName;
    }

    namePosition = Camera.main.WorldToScreenPoint (new
Vector3(this.transform.position.x, this.transform.position.y+1.2f,
this.transform.position.z));
}
```

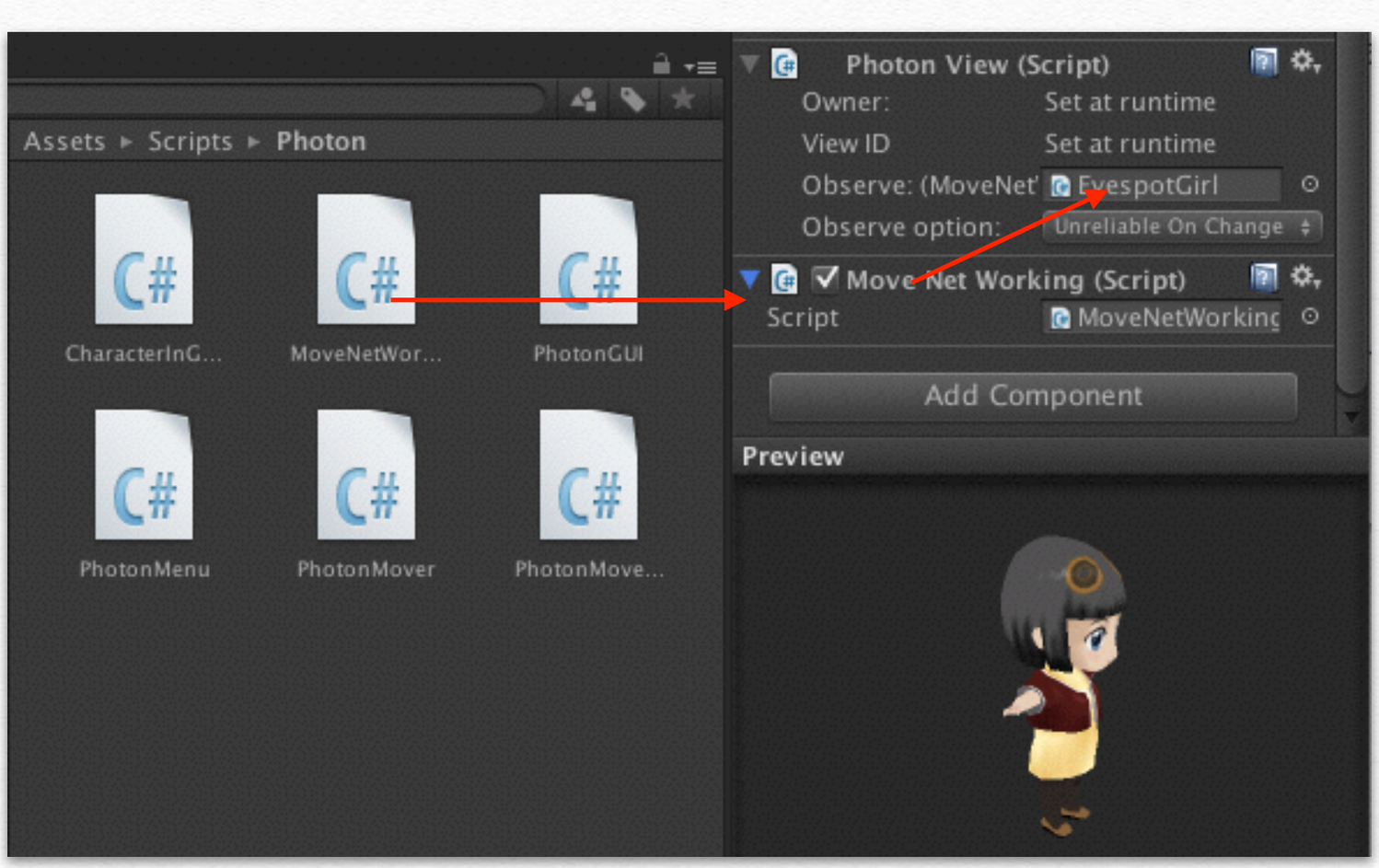
顯示GUI

- 用GUI將玩家暱稱顯示於頭頂上

```
    void OnGUI () {
        GUIStyle LabelStyle = new GUIStyle(GUI.skin.label);

        LabelStyle.alignment = TextAnchor.MiddleCenter;
        LabelStyle.fontStyle = FontStyle.Bold;
        LabelStyle.normal.textColor = Color.white;


        GUI.Label (new Rect (namePosition.x - 50, Screen.height -
namePosition.y, 100, 20), characterName, LabelStyle);
    }
```

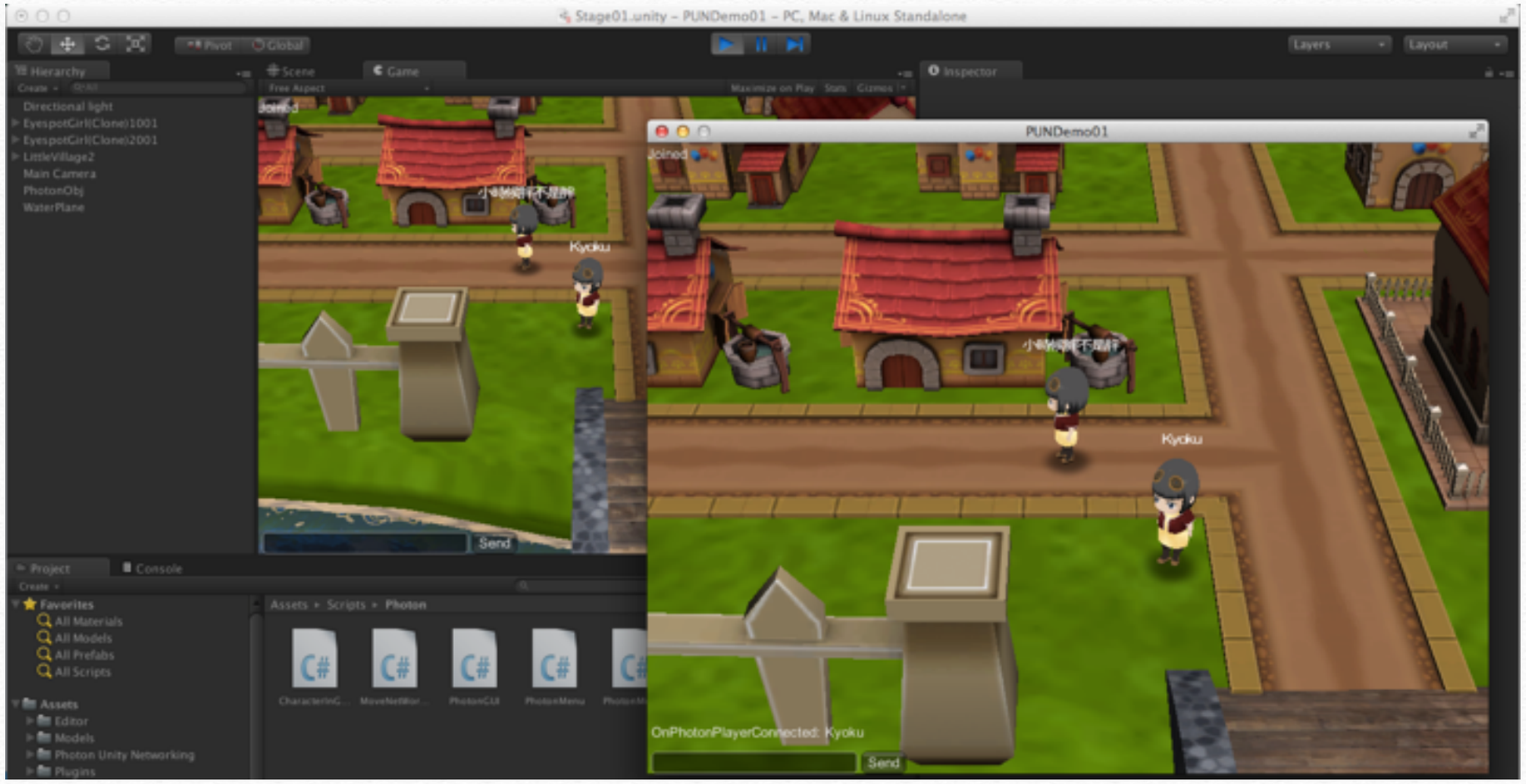
將腳本加到Prefab上

- 將腳本加到角色Prefab上，並將MoveNetWorking拉到PhotonView的Observe上，Opserve option設為Reliable Delta Compressed



編譯後測試

- 將遊編譯後進行測試



進一步最佳化

- 讀者應該會發現，所有的玩家會從原點出現後滑到定位點，看起來不太自然，因為腳色建立之後才會廣播位置因此會有此問題，可自行建立旗標，先隱藏角色，若第一次收到廣播則直接角色定到目標點，接著才顯示出角色。因為這些已是基本邏輯，請讀者自行處理。

End